UNIT-4 Basic Methods in Supervised Learning

Distance -based methods, Nearest- Neighbors, Decision Trees, Support Vector Machines, Nonlinearity and Kernel Methods. Unsupervised Learning: Clustering, K-means, Dimensionality Reduction. PCA and kernel

1. Supervised Learning

Definition:

Supervised learning is a type of machine learning where the model learns from labeled training data. The goal is to map inputs (features) to correct outputs (labels).

Examples:

- **Email spam classification** (Spam or Not Spam)
- **Handwritten digit recognition** (0-9)
- **Predicting house prices** based on area and number of rooms

1.1 Distance-Based Methods

Concept:

Distance-based methods classify data points based on similarity, measured using distance metrics.

Common Distance Metrics

1. **Euclidean Distance**

   Measures the straight-line distance between two points:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

2. **Manhattan Distance**

   Measures distance along axes at right angles:

$$d(A,B) = \sum_{i=1}^{n} |A_i - B_i|$$

3. **Minkowski Distance** (Generalization of Euclidean & Manhattan)

$$d(A,B) = \left(\sum_{i=1}^{n} |A_i - B_i|^p\right)^{\frac{1}{p}}$$

- **p = 1** → Manhattan Distance
- **p = 2** → Euclidean Distance

Used in **K-Nearest Neighbors (KNN)** and **clustering algorithms**.

1.2 Nearest-Neighbors (K-Nearest Neighbors - KNN)

Concept:

- A **non-parametric, instance-based learning algorithm**.
- A new data point is classified based on the **majority class of its K-nearest neighbors**.

Steps in KNN:

1. Choose **K** (number of neighbors).
2. Compute the distance between the new point and existing data points.
3. Select the **K closest neighbors**.
4. Assign the most common label (majority vote in classification).

Example:

Predict if a **customer will buy a product** based on past customer behavior.

✅ **Pros:** Simple, effective, non-parametric.
❌ **Cons:** Slow for large datasets, sensitive to irrelevant features.

1.3 Decision Trees

Concept:
A hierarchical structure that makes decisions based on feature conditions.
Each node represents a **feature**, branches represent **decisions**, and leaves represent **final outputs**.

Key Features:

- Uses **Entropy and Information Gain** to determine the best feature.
- Works for **classification** and **regression**.

Example Decision Tree for Weather Prediction

Outlook	Humidity	Wind	Play?
Sunny	High	Weak	No
Overcast	Normal	Strong	Yes

The tree splits on the most **important feature first** to reduce uncertainty.

✅ **Pros:** Interpretable, fast, handles mixed data.
❌ **Cons:** Can overfit, sensitive to noisy data.

Advanced Versions:

- **Random Forest** (ensemble of decision trees)
- **Gradient Boosted Trees** (boosting-based improvement)

1.4 Support Vector Machines (SVM)

Concept:

- A supervised learning algorithm used for **classification and regression**.
- Finds a **hyperplane** that best separates classes in a high-dimensional space.

Key Concepts:

1. **Margin:** Distance between the hyperplane and the closest data points (support vectors).
2. **Kernel Trick:** Allows SVM to work with non-linearly separable data by transforming input space.

Example:

- **Linear SVM** → Used for linearly separable data.
- **Kernel SVM (RBF, Polynomial, Sigmoid)** → Used for complex data.

✅ **Pros:** Works well on small datasets, robust against overfitting.
❌ **Cons:** Computationally expensive for large datasets.

1.5 Nonlinearity and Kernel Methods

Concept:

- Many real-world problems are **non-linear**, meaning they can't be separated using a straight line.
- **Kernel methods** transform data into higher-dimensional space to make it separable.

Common Kernels:

1. **Polynomial Kernel:**

$$K(x, y) = (x \cdot y + c)^d$$

2. **Radial Basis Function (RBF) Kernel:**

$$K(x, y) = e^{-\gamma||x-y||^2}$$

✅ **Pros:** Solves complex classification problems.
❌ **Cons:** Requires careful hyperparameter tuning.

---

2. Unsupervised Learning

Definition:

- Unsupervised learning deals with **unlabeled data**.
- The goal is to find **patterns, clusters, or compressed representations**.

Examples:

- **Customer segmentation**
- **Anomaly detection**
- **Topic modeling in NLP**

2.1 Clustering

Clustering groups similar data points together.

2.1.1 K-Means Clustering

Algorithm:

1. Choose **K** (number of clusters).
2. Initialize **K random centroids**.
3. Assign each point to the **nearest centroid**.
4. Update centroids by computing the mean.
5. Repeat until centroids **no longer change**.

Example:

- Segmentation of customers into high-value, medium-value, and low-value groups.

✅ **Pros:** Simple, fast, scalable.
❌ **Cons:** Requires K to be predefined, sensitive to initial centroids.

2.2 Dimensionality Reduction

- **High-dimensional data** (many features) can be hard to process.
- **Dimensionality reduction** reduces the number of features while preserving **important information**.

2.2.1 Principal Component Analysis (PCA)

- **Concept:**
  - PCA transforms data into a new coordinate system, maximizing variance.
  - Finds **principal components** that explain the most variance.

Steps:

1. Compute the **covariance matrix**.
2. Compute **eigenvectors and eigenvalues**.
3. Choose **top K eigenvectors** as new feature space.

✅ **Pros:** Reduces dimensionality, removes noise.
❌ **Cons:** Can lose interpretability.

2.3 Kernel PCA

- PCA assumes linear relationships.
- **Kernel PCA** applies **non-linear transformations** before PCA.
- **Example:**
  - Recognizing handwritten digits.

✅ **Pros:** Works for non-linear data.
❌ **Cons:** Computationally expensive.

Summary Table

Method	Type	Used For	Key Feature
KNN	Supervised	Classification	Distance-based, instance learning
Decision Tree	Supervised	Classification/Regression	Splits based on feature importance
SVM	Supervised	Classification	Finds optimal hyperplane
K-Means	Unsupervised	Clustering	Groups similar data
PCA	Unsupervised	Dimensionality Reduction	Finds principal components
Kernel PCA	Unsupervised	Non-linear Dimensionality Reduction	Uses kernel tricks

---

--:-:--

1.What is supervised learning? Give two examples.

Supervised Learning: Detailed Explanation

What is Supervised Learning?

Supervised learning is a type of **machine learning** where a model is trained using **labeled data** (i.e., each input has a corresponding output label). The goal is to learn a mapping from input variables (**X**) to output variables (**Y**) so the model can make predictions on unseen data.

- **Supervised learning tasks are divided into two main types:**
  - **Classification:** Predicting a categorical output (e.g., spam vs. not spam, disease diagnosis).
  - **Regression:** Predicting a continuous numerical output (e.g., house price prediction, stock market trends).

1. Spam Email Detection (Classification)

Problem Statement:
Given a dataset of emails, classify whether an email is **Spam (1)** or **Not Spam (0)** based on certain features like subject line, words in the email, and sender details.

How It Works:

1. **Dataset:**
    - The training data consists of emails labeled as **Spam (1)** or **Not Spam (0)**.
    - Features may include **word frequency, presence of certain keywords (e.g., "free," "win"), sender address, and email structure**.
2. **Training the Model:**
    - The model learns from past email patterns using a classification algorithm like **Naïve Bayes Classifier** or **Decision Trees**.
    - It identifies patterns in spam emails, such as repeated words like **"lottery," "prize," "free," or "urgent"**.
3. **Prediction on New Emails:**
    - The trained model checks new emails and predicts whether they are **spam or not spam** based on learned features.
4. **Real-World Example:**
    - Email services like **Gmail, Yahoo, and Outlook** use supervised learning to filter spam emails.

---

2. House Price Prediction (Regression)

Problem Statement:
Predict the price of a house based on features such as **square footage, number of bedrooms, location, and age of the house**.

How It Works:

1. **Dataset:**
    - The training data contains historical house prices along with corresponding features.
    - Example data:

Square Footage	Bedrooms	Location	Age (Years)	Price (in $)
2000	3	City A	5	250,000
1500	2	City B	10	180,000

2. **Training the Model:**
   - The model uses **Linear Regression** or **Decision Trees** to identify how features affect house price.
   - It learns that **larger square footage and better location lead to higher prices**.
3. **Prediction on New Houses:**
   - Given a new house's features, the model predicts the expected price.
4. **Real-World Example:**
   - Real estate platforms like **Zillow, Redfin, and Housing.com** use supervised learning to estimate property prices.

Common Supervised Learning Algorithms

Algorithm	Type	Usage
Naïve Bayes	Classification	Spam filtering, sentiment analysis
Decision Trees	Classification & Regression	Loan approvals, medical diagnosis
Support Vector Machines (SVM)	Classification	Face recognition, fraud detection
K-Nearest Neighbors (KNN)	Classification & Regression	Recommendation systems, pattern recognition
Linear Regression	Regression	House price prediction, stock prices
Random Forest	Classification & Regression	Fraud detection, customer segmentation

---

2.Define the K-Nearest Neighbors (KNN) algorithm.

K-Nearest Neighbors (KNN) Algorithm: Explanation

Definition:

K-Nearest Neighbors (KNN) is a **supervised learning algorithm** used for both **classification** and **regression** tasks. It is a **distance-based** algorithm that classifies new data points based on the majority class of their **k nearest neighbors** in the dataset.

How KNN Works:

1. **Store Training Data:**
   - KNN **does not explicitly train a model**; instead, it stores all training data.

2. **Calculate Distance:**
   - When a new data point is given, KNN calculates its distance from all existing data points using distance metrics like:
     - **Euclidean Distance** (most common)
     - Manhattan Distance
     - Minkowski Distance
3. **Find Nearest Neighbors:**
   - Select the **k** closest data points (neighbors) to the new data point.
4. **Majority Voting (Classification) or Averaging (Regression):**
   - **Classification:** Assigns the most common class among the **k neighbors** to the new data point.
   - **Regression:** Takes the average value of **k nearest neighbors** as the predicted value.

Example of KNN for Classification

Problem: Classify whether a fruit is an **apple or orange** based on **weight and texture.**

- **Dataset:**

Weight (grams)	Texture	Fruit
150	Smooth	Apple
180	Rough	Orange
130	Smooth	Apple
160	Rough	Orange

- **New Data:** Given a fruit with **weight = 140g** and **texture = Smooth**, classify it.

Steps:

1. Calculate the distance of the new data point from each fruit in the dataset.
2. Select the **k nearest neighbors** (e.g., k = 3).
3. **Majority voting**: If 2 out of 3 neighbors are **Apples**, classify the new fruit as an **Apple**.

Example of KNN for Regression

Problem: Predict the price of a house based on similar houses' prices.

- **Dataset:**

Square Footage	Bedrooms	Price (in $)
1500	3	250,000
1800	4	300,000
1200	2	200,000

- **New House:** Given a house with **1600 sq. ft., 3 bedrooms**, predict the price.

Steps:

1. Find the **k nearest houses**.
2. Take the **average price** of these houses to predict the price of the new house.

Advantages of KNN

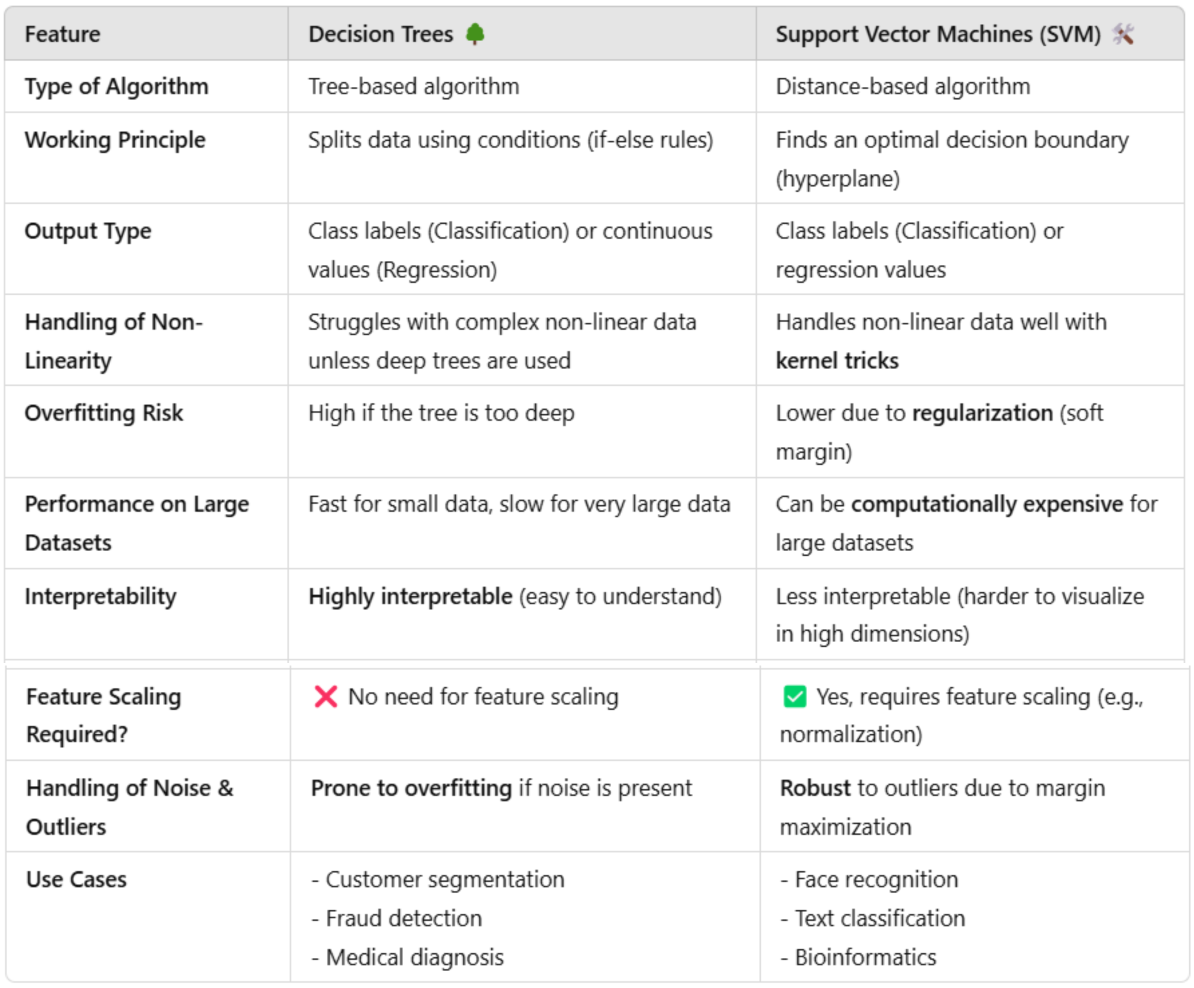
✅ **Simple & Easy to Implement**
✅ **Works for Classification & Regression**
✅ **No Training Required** (Instance-based learning)

Disadvantages of KNN

❌ **Slow for Large Datasets** (Computation increases with more data)
❌ **Sensitive to Irrelevant Features** (Feature selection is important)
❌ **Choosing the Right Value of k is Crucial**

3.List the key differences between Decision Trees and Support Vector Machines (SVM).

Key Differences Between Decision Trees and Support Vector Machines (SVM)



Feature	Decision Trees 🌳	Support Vector Machines (SVM) 🛠️
Type of Algorithm	Tree-based algorithm	Distance-based algorithm
Working Principle	Splits data using conditions (if-else rules)	Finds an optimal decision boundary (hyperplane)
Output Type	Class labels (Classification) or continuous values (Regression)	Class labels (Classification) or regression values
Handling of Non-Linearity	Struggles with complex non-linear data unless deep trees are used	Handles non-linear data well with **kernel tricks**
Overfitting Risk	High if the tree is too deep	Lower due to **regularization** (soft margin)
Performance on Large Datasets	Fast for small data, slow for very large data	Can be **computationally expensive** for large datasets
Interpretability	**Highly interpretable** (easy to understand)	Less interpretable (harder to visualize in high dimensions)
Feature Scaling Required?	❌ No need for feature scaling	✅ Yes, requires feature scaling (e.g., normalization)
Handling of Noise & Outliers	**Prone to overfitting** if noise is present	**Robust** to outliers due to margin maximization
Use Cases	- Customer segmentation - Fraud detection - Medical diagnosis	- Face recognition - Text classification - Bioinformatics

---

Summary

📌 **Decision Trees** are **rule-based models**, easy to interpret, but prone to overfitting.
📌 **SVM** finds the **best boundary** to separate classes, handles non-linearity well, but is computationally expensive.

4.What is the purpose of the kernel trick in machine learning?

Purpose of the Kernel Trick in Machine Learning

The **kernel trick** is a technique used in machine learning, particularly in **Support Vector Machines (SVMs)** and other algorithms, to handle **non-linearly separable data**. It allows models to work in **higher-dimensional feature spaces** without explicitly computing those dimensions.

Why Do We Need the Kernel Trick?

Many real-world datasets are **not linearly separable** in their original feature space. Instead of manually adding new features, the kernel trick **implicitly transforms the data** into a higher-dimensional space where it becomes separable.

How Does the Kernel Trick Work?

- The **original data** exists in a lower-dimensional space.
- A **kernel function** transforms it into a higher-dimensional space.
- The model then finds a **linear decision boundary** in the higher-dimensional space, which translates into a **non-linear boundary** in the original space.

📌 **Key Advantage:**
Instead of **explicitly computing** the transformation (which can be computationally expensive), the kernel trick allows us to **compute inner products** in the higher-dimensional space **efficiently** using kernel functions.

Common Kernel Functions

1️⃣ **Linear Kernel:**

$$K(x, y) = x^T y$$

- Used when data is **already linearly separable.**

2️⃣ **Polynomial Kernel:**

$$K(x, y) = (x^T y + c)^d$$

- Used for **curved decision boundaries.**

3️⃣ **Radial Basis Function (RBF) Kernel** (Gaussian Kernel):

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

- Most commonly used kernel in SVM.
- Good for **complex and highly non-linear data**.

4️⃣ **Sigmoid Kernel**:

$$K(x, y) = \tanh(\alpha x^T y + c)$$

- Used in **neural networks**.

Example Use Case:

Imagine you have a dataset where points from **two classes** are arranged in a circular pattern. A **linear classifier** (like Logistic Regression or Linear SVM) would **fail** to separate them.

🔹 **Without Kernel Trick**: A straight line cannot separate the classes.

🔹 **With Kernel Trick (RBF Kernel)**: The data is mapped to a higher-dimensional space where a **hyperplane** (linear separator) can distinguish them.

5.Define clustering and explain its importance in unsupervised learning.

Definition of Clustering

Clustering is a technique in **unsupervised learning** where data points are grouped into clusters based on their similarity. The goal is to divide the dataset into **homogeneous groups** such that:

- Data points **within the same cluster** are more similar to each other.
- Data points in **different clusters** are as dissimilar as possible.

📌 **Key Idea:** There are **no predefined labels**, and the algorithm finds patterns or structures in the data.

Importance of Clustering in Unsupervised Learning

Clustering is one of the most fundamental tasks in **unsupervised learning** and has several **practical applications**:

1️⃣ **Pattern Recognition**

- Identifies hidden patterns in data, which is useful for **data exploration**.

2️⃣ **Customer Segmentation**

- Businesses group customers based on **purchasing behavior** and preferences for **targeted marketing**.

3️⃣ **Anomaly Detection**

- Detects **fraudulent transactions**, **network intrusions**, and **manufacturing defects** by identifying outliers.

4️⃣ **Recommendation Systems**

- Used in e-commerce and streaming services to suggest products/movies based on similar users' behavior.

5️⃣ **Image Segmentation**

- Divides an image into different **regions** (e.g., background, objects) in computer vision applications.

6️⃣ **Genomics & Healthcare**

- Identifies groups of **genes with similar expressions** or clusters **patients with similar symptoms** for personalized medicine.

Types of Clustering Algorithms

There are different types of clustering algorithms, each with its strengths:

Clustering Type	Example Algorithms	Description
Partition-Based	**K-Means, K-Medoids**	Groups data into `K` predefined clusters.
Density-Based	**DBSCAN, OPTICS**	Finds clusters based on dense regions in data.
Hierarchical-Based	**Agglomerative, Divisive**	Builds a tree-like structure of clusters.
Model-Based	**Gaussian Mixture Model (GMM)**	Assumes data is generated from a mixture of distributions.

Example: Clustering Customers Using K-Means

Suppose a retail company wants to **segment its customers** based on their spending patterns.

1️⃣ Collect **purchase history data** (e.g., frequency, amount spent).
2️⃣ Apply **K-Means Clustering** to divide customers into groups.
3️⃣ Identify **customer categories** (e.g., High Spenders, Occasional Buyers, Discount Seekers).
4️⃣ Use insights for **targeted marketing** and personalized recommendations.

6.Explain how the Euclidean distance is used in K-Nearest Neighbors (KNN).

Euclidean Distance in K-Nearest Neighbors (KNN)

The **K-Nearest Neighbors (KNN)** algorithm is a distance-based machine learning method used for **classification and regression**. It works by finding the **K closest data points (neighbors)** to a new data point and using their labels to make a prediction.

One of the most commonly used distance metrics in KNN is **Euclidean Distance**.

What is Euclidean Distance?

Euclidean Distance is the straight-line distance between two points in a multidimensional space. It is calculated using the formula:

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

For **higher dimensions (n features):**

$$d(A,B) = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2}$$

Where:

- X_i and Y_i are the feature values of two data points.
- $d(A,B)$ is the Euclidean distance between points A and B.

How is Euclidean Distance Used in KNN?

1️⃣ Calculate Distances

- Given a new data point, compute the **Euclidean distance** between it and all points in the dataset.

2️⃣ Find the K-Nearest Neighbors

- Select the **K smallest distances**, i.e., the closest points to the new data point.

3️⃣ Make a Prediction

- **For classification**: Assign the most common class label among the K neighbors.
- **For regression**: Take the average value of the K neighbors.

Example: KNN with Euclidean Distance

Problem:

We have the following dataset of students categorized as "Pass" or "Fail" based on study hours and sleep hours.

Study Hours	Sleep Hours	Result
4	6	Pass
2	8	Fail
5	5	Pass
1	9	Fail
6	4	Pass

Now, for a new student who studied **3 hours** and slept **7 hours**, we want to predict whether they will **Pass or Fail** using KNN (**K=3**).

Step 1: Compute Euclidean Distance

Calculate the Euclidean distance between the new student **(3,7)** and each existing student:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Existing Student (X, Y)	Distance from (3,7)
(4,6) → Pass	$\sqrt{(4-3)^2 + (6-7)^2} = \sqrt{1+1} = 1.41$
(2,8) → Fail	$\sqrt{(2-3)^2 + (8-7)^2} = \sqrt{1+1} = 1.41$
(5,5) → Pass	$\sqrt{(5-3)^2 + (5-7)^2} = \sqrt{4+4} = 2.83$
(1,9) → Fail	$\sqrt{(1-3)^2 + (9-7)^2} = \sqrt{4+4} = 2.83$
(6,4) → Pass	$\sqrt{(6-3)^2 + (4-7)^2} = \sqrt{9+9} = 4.24$

Step 2: Find the K (3) Nearest Neighbors

The 3 closest points to **(3,7)** are:

1. **(4,6)** → **Pass** (1.41)
2. **(2,8)** → **Fail** (1.41)
3. **(5,5)** → **Pass** (2.83)

Step 3: Assign Class Label

- **2 Pass** and **1 Fail** → Majority class is **Pass**
- Prediction: **The new student is likely to Pass** ✅

7.How does the K-Means clustering algorithm work? Describe the steps involved.

K-Means Clustering Algorithm

K-Means is an **unsupervised learning algorithm** used for **clustering** data into **K distinct groups**. It partitions the dataset into **K clusters**, where each data point belongs to the nearest cluster based on the centroid (mean) of the cluster.

Steps of K-Means Algorithm

Step 1: Choose the Number of Clusters (K)

- Decide the number of clusters K to segment the data.
- The value of K can be chosen using methods like the **Elbow Method** or **Silhouette Score**.

Step 2: Initialize Centroids

- Randomly select K points from the dataset as the initial **cluster centroids**.
- These centroids serve as the center of each cluster.

Step 3: Assign Data Points to the Nearest Centroid

- Calculate the **Euclidean distance** between each data point and all centroids.
- Assign each data point to the **nearest centroid** (i.e., the closest cluster).

Step 4: Compute New Centroids

- For each cluster, compute the **mean (average) position** of all assigned data points.
- The new mean becomes the **new centroid**.

Step 5: Repeat Until Convergence

- Repeat **Step 3 and Step 4** until the centroids **no longer change** significantly or the maximum iterations are reached.
- The clusters are now finalized.

Example of K-Means Clustering

Problem:

Suppose we have a dataset of customer spending behavior in a shopping mall. We want to segment customers into **3 groups (K=3)** based on their **annual income** and **spending score**.

Dataset (Sample Data Points)

Customer ID	Annual Income ($)	Spending Score
1	15,000	40
2	18,000	42
3	30,000	80
4	35,000	85
5	50,000	30

Step-by-Step Execution

1️⃣ **Initialize K=3 centroids randomly** (e.g., three initial customer locations).

2️⃣ **Assign each customer to the nearest centroid** based on **Euclidean Distance**.

3️⃣ **Compute new centroids** (average of assigned customers).

4️⃣ **Repeat until centroids no longer change** significantly.

8.Compare and contrast Principal Component Analysis (PCA) and Kernel PCA.

Comparison of Principal Component Analysis (PCA) and Kernel PCA

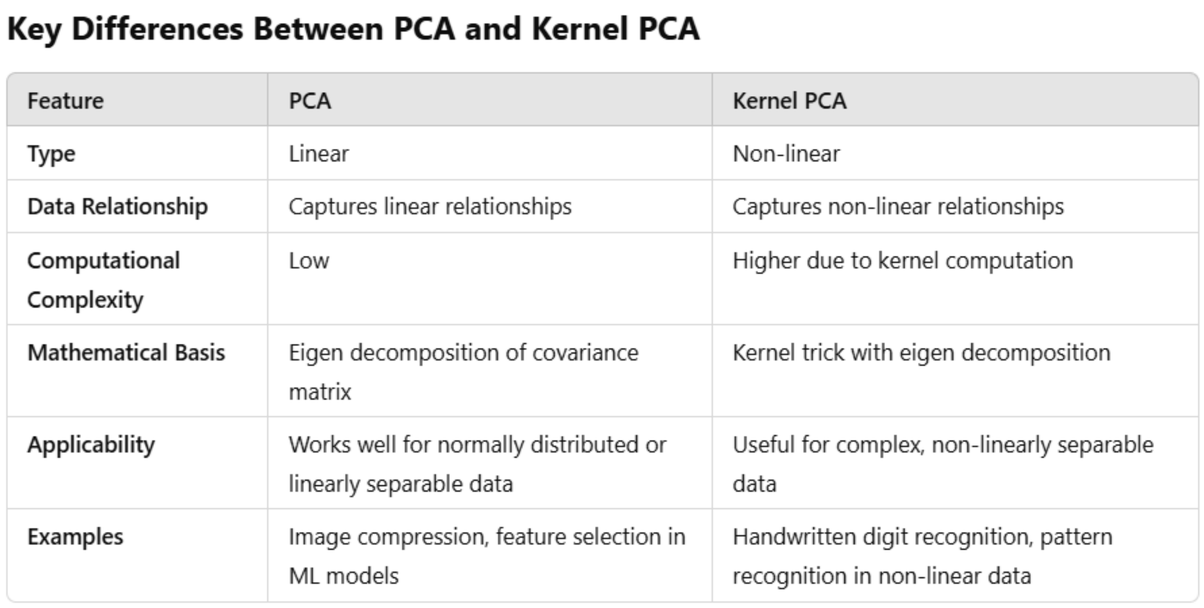
1. Principal Component Analysis (PCA)

- PCA is a **linear dimensionality reduction technique** that finds the directions (principal components) of maximum variance in high-dimensional data.
- It projects the data onto a lower-dimensional space while preserving as much variance as possible.

- Uses **Eigenvalue decomposition** or **Singular Value Decomposition (SVD)** to compute principal components.
- PCA works best when the data is linearly separable.
- Computationally efficient, but limited in handling complex patterns.

2. Kernel Principal Component Analysis (Kernel PCA)

- Kernel PCA is an **extension of PCA** that enables **non-linear dimensionality reduction** using the **kernel trick**.
- It maps data into a higher-dimensional feature space where linear PCA is applied.
- Common kernels include **Radial Basis Function (RBF), Polynomial, and Sigmoid kernels**.
- More computationally expensive than PCA but captures complex, non-linear patterns.
- Useful in scenarios where the relationship between variables is non-linear.



Key Differences Between PCA and Kernel PCA

Feature	PCA	Kernel PCA
Type	Linear	Non-linear
Data Relationship	Captures linear relationships	Captures non-linear relationships
Computational Complexity	Low	Higher due to kernel computation
Mathematical Basis	Eigen decomposition of covariance matrix	Kernel trick with eigen decomposition
Applicability	Works well for normally distributed or linearly separable data	Useful for complex, non-linearly separable data
Examples	Image compression, feature selection in ML models	Handwritten digit recognition, pattern recognition in non-linear data

9.Suppose you have a dataset of customer transactions. How would you use Decision Trees to classify customers into different spending categories?

Using Decision Trees to Classify Customers into Spending Categories

A **Decision Tree** is a supervised learning algorithm used for classification and regression tasks. It works by recursively splitting the dataset into subsets based on feature values, forming a tree-like structure to make decisions.

Steps to Use Decision Trees for Customer Spending Classification

1. Define the Problem

- Suppose we have a dataset of customer transactions, including features like:
  - **Age**
  - **Income**
  - **Transaction Frequency**
  - **Total Amount Spent**
  - **Purchase Categories (Electronics, Groceries, Fashion, etc.)**
- The goal is to classify customers into different **spending categories** (e.g., **Low, Medium, High**).

---

2. Data Collection & Preprocessing

- **Gather the dataset** from purchase records.
- **Handle missing values** (e.g., using mean/mode imputation).
- **Encode categorical variables** (e.g., one-hot encoding for purchase categories).
- **Normalize numerical features** (e.g., Min-Max Scaling for income and spending amounts).

---

3. Feature Selection

- Choose relevant features such as:
  - **Customer Age**
  - **Annual Income**
  - **Average Monthly Spending**
  - **Transaction Frequency**

These features help the decision tree determine the spending category.

---

4. Building the Decision Tree Model

- **Split the dataset** into **training (80%)** and **testing (20%)** sets.
- Train a **Decision Tree Classifier** using **Gini Impurity** or **Entropy** as a splitting criterion.

5. Decision Tree Structure & Interpretability

- The Decision Tree splits data based on feature importance.
- Example of Decision Tree Splitting:

- **If Income > 50K:** Check Transaction Frequency
- **If Transaction Frequency > 20/month:** High spender
- **Else:** Medium spender
- **If Income < 50K:** Low spender

6. Model Optimization

- **Pruning**: Reduce tree complexity to avoid overfitting.
- **Hyperparameter Tuning**: Adjust max_depth, min_samples_split, etc.
- **Cross-validation**: Improve model generalization.

10.Given a dataset with high-dimensional features, explain how PCA can be applied to reduce dimensionality while preserving important information.

Principal Component Analysis (PCA) for Dimensionality Reduction

1. Introduction to PCA

Principal Component Analysis (PCA) is a **dimensionality reduction** technique that transforms a high-dimensional dataset into a lower-dimensional space while **preserving the most important information**. It helps in reducing **computational cost**, avoiding **overfitting**, and improving the performance of machine learning models.

---

2. Why Use PCA for High-Dimensional Data?

- High-dimensional data can be **redundant** or contain **correlated features**.
- Machine learning models suffer from the **curse of dimensionality** (more features → more computation & risk of overfitting).
- PCA identifies the **most important directions** (principal components) in the data.

3. Steps to Apply PCA for Dimensionality Reduction

Step 1: Standardize the Data

Since PCA relies on variance, the data needs to be scaled to have **zero mean and unit variance**.

Formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

where:

- X is the original feature,
- μ is the mean,
- σ is the standard deviation.

Step 2: Compute the Covariance Matrix

The covariance matrix shows relationships between different features.

$$C = \frac{1}{n}(X^T X)$$

where X is the standardized data matrix.

Step 3: Compute Eigenvalues & Eigenvectors

- **Eigenvalues** determine the variance explained by each principal component.
- **Eigenvectors** represent the new feature space (principal components).

Step 4: Select Principal Components

- Choose the top **k** components that capture the most variance.
- The sum of their explained variance should be ≥ **95%** of total variance.

Step 5: Transform Data into the New Feature Space

$$Z = X \cdot W$$

where:

- X is the original dataset,
- W is the matrix of selected eigenvectors (principal components).

---

5. Benefits of PCA

✅ **Reduces computation time** for high-dimensional datasets.
✅ **Removes correlated features**, improving model efficiency.
✅ **Helps visualization** of high-dimensional data.

However, PCA **loses some information** because it's a lossy transformation.
